

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Marolt

# **Diagnostika avtomobila z mikrokrmilnikom Arduino**

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM  
PRVE STOPNJE  
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Branko Šter

Ljubljana, 2017

COPYRIGHT. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

*Besedilo je oblikovano z urejevalnikom besedil  $\text{\LaTeX}$ .*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Povežite mikrokrmilnik Arduino z avtomobilskim računalnikom preko vmesnika OBD2. Mikrokrmilnik naj poskrbi za povezavo in prikaz različnih uporabnih parametrov avtomobila, podatke pa naj shranjuje na pomnilno kartico SD za kasnejši ogled na računalniku.



*Zahvaljujem se mentorju, prof. dr. Branku Šteru, za podporo in nasvete pri izdelavi diplomske naloge. Posebna zahvala pa gre še mojemu dekletu in družini za podporo pri študiju.*



# Kazalo

**Povzetek**

**Abstract**

<b>1</b>	<b>Uvod</b>	<b>1</b>
1.1	Pregled področja . . . . .	2
<b>2</b>	<b>Opis uporabljenih orodij, naprav in tehnologij</b>	<b>5</b>
2.1	OBD . . . . .	5
2.2	Vmesnik ELM 327 . . . . .	6
2.3	Arduino . . . . .	7
2.4	Arduino IDE . . . . .	9
<b>3</b>	<b>Razvoj strojne opreme</b>	<b>11</b>
3.1	BlueTooth modul HC-05 . . . . .	11
3.2	LCD zaslon . . . . .	14
3.3	SD bralnik . . . . .	16
3.4	Realna ura . . . . .	18
<b>4</b>	<b>Razvoj programske opreme</b>	<b>21</b>
4.1	Konfiguriranje vmesnika ELM 327 . . . . .	21
4.2	Seznam uporabljenih parametrov . . . . .	25
4.3	Izračun trenutne porabe . . . . .	27
4.4	Pomembni deli kode . . . . .	29

<b>5</b>	<b>Testiranje</b>	<b>33</b>
5.1	Končni izdelek . . . . .	33
5.2	Testiranje na avtomobilih . . . . .	35
<b>6</b>	<b>Sklepne ugotovitve</b>	<b>39</b>
	<b>Literatura</b>	<b>42</b>



# Seznam uporabljenih kratic

kratica	angleško	slovensko
<b>OBD</b>	On-board diagnostics	Diagnostika v avtomobilu
<b>OBD2</b>	On-board diagnostics 2	Izboljšana verzija OBD
<b>ECU</b>	Engine Control Unit	Nadzorni računalnik v avtomobilu
<b>SD</b>	Secure Digital	Tip pomnilne kartice
<b>SPI</b>	Serial Peripheral Interface Bus	Tip serijskega vodila
<b>I<sup>2</sup>C</b>	Inter-Integrated Circuit	Tip serijskega vodila
<b>LCD</b>	Liquid Cristal Display	Zaslون iz tekočih kristalov
<b>PID</b>	Parameter ID	Identifikator parametra
<b>MIL</b>	Malfunction indicator lamp	Lučka za napako



# Povzetek

**Naslov:** Diagnostika avtomobila z mikrokrmilnikom Arduino

**Avtor:** Blaž Marolt

V Evropi je že od leta 2001 v avtomobilih po zakonu uporabljen protokol OBD2, ki je standarden za vse avtomobilske proizvajalce. Protokol je bil zasnovan leta 1969, predstavljen pa leta 1980. Takrat je omogočal samo to, da so lahko mehaniki brali statusne različnih podsistemov v avtomobilu. Od takrat naprej je njegova uporabnost naraščala do te mere, da lahko danes prikazujemo različne parametre avtomobila v realnem času, beremo zabeležene napake, jih brišemo in spreminjamo. Veliko teh parametrov lahko prikazujemo na potovalnem računalniku, vendar tega pri starejših avtomobilih ni. Cilj diplomske naloge je uporabiti mikrokrmilnik Arduino in vmesnik OBD2 ter z njima prikazovati in shranjevati parametre avtomobila. Končni izdelek bo montiran v avtomobil.

**Ključne besede:** OBD, avtomobil, diagnostika avtomobila, Arduino.



# Abstract

**Title:** Car diagnostics with microcontroller Arduino

**Author:** Blaž Marolt

Since 2001 in Europe cars use OBD2 protocol, which is standard for all automotive manufacturers. The protocol was designed in 1969 and introduced in 1980. At that time, it only allowed the mechanics to read the statuses of various subsystems in the car. Since then, its use increased to such extent that today we can show different parameters of a car in real time, read the recorded errors, as well as delete and modify them. Many of these parameters can be shown on the trip computer, but not in older cars. The aim of this thesis is to use the Arduino microcontroller and OBD2 interface and use them to read and save the car's parameters. The final product will be installed in a car.

**Keywords:** OBD, car, car diagnostics, Arduino.



# Poglavje 1

## Uvod

Avtomobili so bili na začetku popolnoma mehanični, vendar je bilo ob napakah v delovanju težko ugotoviti, kaj je narobe, ne da bi jih razstavili in fizično pogledali. Zato so ob porastu elektronike v avtomobile začeli vgrajevati elektronske komponente, aktuatorje in senzorje za nadzor za pomoč pri diagnosticiranju napak. Volkswagen je že leta 1969 poskusno začel uporabljati sistem za branje napak pri njihovih modelih. Leta 1980 pa je General Motors predstavil sistem, ki je bil vgrajen v njihova vozila v ZDA. S tem so lahko izbrisali lučko za napako na motorju in brali kode napak. Sistem je bil počasen, poleg tega pa omejen na njihova vozila. Vlada ZDA je leta 1991 zahtevala izdelavo standarda avtomobilske diagnostike, ki bi bila ista za vse proizvajalce avtomobilov. Takrat so izdali najprej protokol OBD, potem pa še njegovo hitrejšo različico OBD2, ki je bil po zakonu obvezen za vse avtomobile, ki so se prodali v ZDA. V Evropi je bil standard OBD2 obvezen leta 2001 za bencinske avtomobile, za dizelske pa šele leta 2003, vendar je večina proizvajalcev te sisteme že prej vgrajevala v avtomobile, tudi zaradi lažjega servisa vozil [11].

Za branje podatkov prek protokola OBD2 potrebujemo napravo za branje. Obstaja veliko naprav, ki nam to omogočajo. Vsi avtomobilski servisi imajo take naprave, vendar so te specialne za znamko avtomobila, ki jo servisirajo, in so zelo drage. Obstaja tudi veliko univerzalnih naprav, ki se priključijo na

OBD priključek (konektor) v avtomobilu in nato omogočajo branje podatkov preko USB, WiFi ali Bluetooth povezave. Sam sem za povezavo uporabil univerzalni čitalec OBD, in sicer ELM 327, na katerega se mikrokrmilnik Arduino poveže preko povezave Bluetooth. Arduino nato zahteva parametre, dekodira odgovor in ga prikaže na zaslonu, po različnih parametrih pa se lahko premikamo s pritiskom na gumb.

Programiranje mikrokrmilnika je potekalo preko Arduino IDE, ki je namensko razvojno okolje. Končni izdelek bo vgrajen v avtomobil in se bo napajal neposredno iz avtomobila. Njegov poglavitni namen je pomoč vozniku pri vožnji in spremljanju delovanja avtomobila, predvsem za tiste voznike, ki si lastijo starejši avtomobil brez potovalnega računalnika, ali pa za bolj napredne voznike, ki jih zanima več o delovanju avtomobila.

V nadaljevanju je najprej predstavljen mikrokrmilnik Arduino, Bluetooth modul, OBD čitalec na osnovi ELM 327, nato postopek vezave in konfiguriranja. Za tem sledi programiranje v Arduino IDE in testiranje končne verzije.

## 1.1 Pregled področja

S to temo so se v nekaterih diplomskih delih ukvarjali tudi na naši fakulteti. Primer tega je diploma Aplikacija za optimizacijo sistema varčne vožnje vozil na motorni pogon [21]. V diplomski nalogi je študent uporabil OBD2 z Bluetooth vmesnikom, na katerega pa se je povezal s pametnega telefona, ki je pošiljal ukaze in prikazoval podatke. Drugi primer je poročilo [22], kjer so ravno tako uporabili OBD2 konektor, vendar z neposredno vezavo na mikrokrmilnik Arduino, ki je potem prikazoval 10 različnih parametrov na zaslonu. Poleg diplomskih del obstaja tudi že nekaj aplikacij za pametni telefon, ki izkoriščajo povezavo Bluetooth za branje podatkov iz OBD vmesnika. Najbolj znana je aplikacija Torque [23].



### 1.1.1 Uporaba aplikacije Torque

Aplikacija Torque je najbolj znana aplikacija za diagnostiko avtomobila, razvita za naprave Android in iOS. Aplikacija potrebuje za delovanje vmesnik ELM 327 s povezavo Bluetooth ali WiFi, za naprave iOS pa je za zdaj podprt le vmesnik s povezavo WiFi. Aplikacija se poveže z vmesnikom in začne prikazovati osnovne podatke. Čeprav v teoriji podpira večino možnih parametrov in tudi brisanje lučke MIL, se je v našem primeru izkazalo, da lahko dostopa do istih stvari kot mi z našim izdelkom, vendar zraven izkorišča tudi podatke, ki jih zbira telefon (pospeškometer, nagib, GPS ...), stvari prikazuje lepše, zraven pa ponuja tudi risanje grafov, spreminjanje vmesnika za bolj pregledno opazovanje parametrov (Slika 1.1).



Slika 1.1: Aplikacija Torque



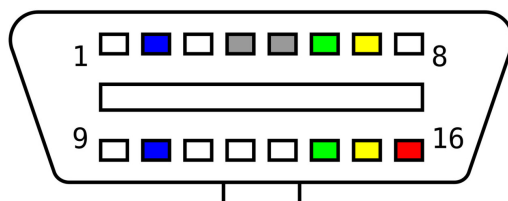
## Poglavje 2

# Opis uporabljenih orodij, naprav in tehnologij

V tem poglavju so na kratko opisane naprave, tehnologije in orodja, ki sem jih uporabil pri izdelavi diplomske naloge.

### 2.1 OBD

OBD (On Board Diagnostics) je zmožnost diagnostike avtomobila z branjem parametrov delovanja avtomobila, s katerim uvidimo, kaj se z avtomobilom dogaja oziroma kaj je z njim narobe. Prvi OBD je Volkswagen razvil že leta 1969 za kasnejše branje napak. Leta 1978 pa so razvili diagnostični računalnik Datsun 280Z, ki je bil že zmožen prikazovati nekaj parametrov v realnem času. General Motors je leta 1980 razvil prvi standard ALDL, ki je bil najprej vgrajen v Kaliforniji, nato pa leta 1981 po celotnih ZDA. Večina uporabnikov je z njim lahko pregledovala in brisala napake. Leta 1991 je vlada ZDA začela zahtevati osnovno diagnostiko avtomobila za vse avtomobile. To se je uresničilo leta 1996, ko so imeli vsi prodani avtomobili že standardiziran konektor, vodila itd., ki so postali znani kot OBD2. V Evropi je standard za diagnostiko za vse nove avtomobile OBD2 postal šele leta 2003. V vseh avtomobilih je tako isti priključek kot na Sliki 2.1.



Slika 2.1: Priključek OBD2

Kljub temu da je OBD2 standard in je zakonsko nujen, uporabljajo proizvajalci avtomobilov drugačne protokole za komunikacijo. Spodaj so naštet različni protokoli [15].

- SAE J1850 PWM  
Uporablja ga Ford, hitrost je 41,6 kbit/s.
- SAE J1850 VPW  
Uporablja ga General Motors, hitrost je 10,4 kbit/s.
- ISO 9141-2  
Uporablja ga večina evropskih in azijskih proizvajalcev, hitrost je 10,4 kbit/s.
- ISO 14230 KWP2000  
Hitrost je 10,4 kbit/s.
- ISO 15765 CAN  
Razvil ga je Bosh v sodelovanju z avtomobilskimi proizvajalci in industrijo. Uporablja se tudi drugje, ne samo v avtomobilski industriji. Hitrost je do 500 kbit/s.

## 2.2 Vmesnik ELM 327

Za komunikacijo na osnovi protokolov OBD2 potrebujemo vmesnik, ki naše ukaze v višjenivojskem jeziku pretvori v nižjenivojski jezik, ki ga uporablja

protokol v določenem avtomobilu. Na začetku je bil najbolj priljubljen vmesnik ELM 327 podjetja ELM electronics. Ker ni bil zaščiten, so vsi kopirali njihovo programsko opremo in jo naložili na svoje cenejše vmesnike. Če sedaj kupimo cenejši vmesnik, verjetno kupimo kitajski klon prvotnega ELM 327. ELM 327 ima lahko vgrajen WiFi, BlueTooth ali USB povezavo za komuniciranje. V tej diplomski nalogi je uporabljen vmesnik ELM 327, kupljen preko interneta, v katerem je po vsej verjetnosti kitajski klon, ima pa povezavo BlueTooth 2.2.



Slika 2.2: Vmesnik ELM 327 s povezavo BlueTooth

## 2.3 Arduino

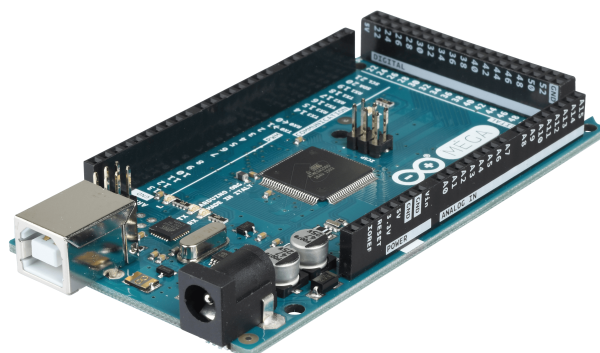
Arduino je odprtokodna elektronska platforma, ki temelji na lahki uporabi strojne in programske opreme. Ponuja nam mnogo različnih modelov razvojnih ploščic, ki jim je skupno to, da vsebujejo mikroprocesor, nekaj digitalnih vhodov in izhodov, analogne vhode in izhode, nekaj pomnilnika za kodo in spremenljivke. Programirajo se v razvojnem okolju Arduino IDE na računalniku, program pa se nato prek USB kabla prenese na izbran mikrokrmilnik, kjer se začne izvajati. Programiranje za mikrokrmilnike Arduino je dobro podprto, z množico že obstoječih knjižic, ki jih lahko vključimo v naš projekt ter veliko že narejenih projektov in primerov [12].

V Tabeli 2.1 so navedeni najbolj pogosti mikrokrmilniki Arduino ter njihovi

najbolj pomembni tehnični podatki:

Tabela 2.1: Primerjava različnih Arduino mikrokrmilnikov

Modul	Analogni V/I	Digitalni VI/PWM	EEPROM[KB]	SRAM[KB]	FLASH[KB]
LilyPad	6/0	14/6	0,5	1	16
Mega 2560	16/0	54/15	4	8	256
Micro	12/0	20/7	1	2,5	32
Pro	6/0	14/6	0,5	1	16
Uno	6/0	14/6	1	2	32
Leonardo	12/0	20/7	1	2,5	32
Nano	8/0	14/6	1	2	32



Slika 2.3: Mikrokrmilnik Arduino Mega



Slika 2.4: Mikrokrmilnik Arduino Uno

Izdelava diplomske naloge je najprej temeljila na mikrokrmilniku Arduino Uno na Sliki 2.4, predvsem zaradi nizke cene tudi na slovenskem tržišču, vendar pa je pred koncem izdelave diplomske naloge prišlo do težav pri pomanjkanju prostora pri pomnilniku FLASH, kamor se shranjuje programska koda, ter pomnilniku SRAM, kamor se shranjujejo spremenljivke [2]. Kljub temu, da sta pomnilnika zasedena samo 85%, nam razvojno okolje Arduino IDE javi, da mikrokrmilniku primanjkuje prostora in da lahko pride do napačnega delovanja. V testiranju smo opazili, da program deluje, če je zasedeno največ 79% prostora; če je bilo zasedenega več, se je program sesul. Zaradi pomanjkanja prostora smo bili tako prisiljeni izbrati drug mikrokrmilnik, zato smo se odločili za Arduino Mega (Slika 2.3), ki ponuja 256 KB pomnilnika FLASH in 8 KB pomnilnika SRAM, kar je dovolj za celoten program.

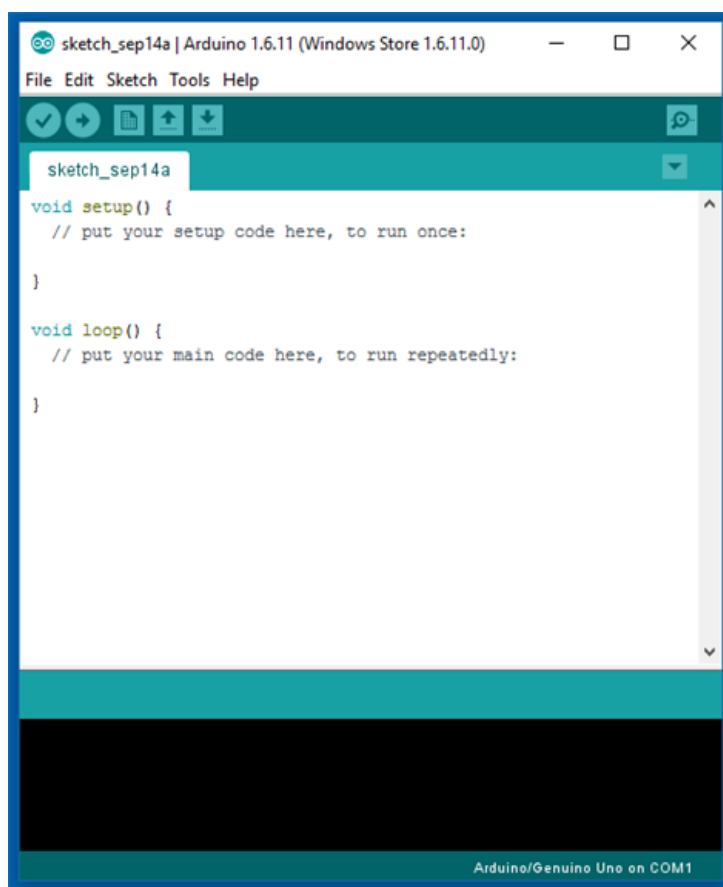
## 2.4 Arduino IDE

Arduino IDE je odprtokodno namensko orodje za programiranje mikrokrmilnika Arduino. Trenutna različica je 1.8.2 in teče na vseh operacijskih sistemih, prav tako pa s tem okoljem lahko programiramo katero koli različico

krmilnika Arduino. Ogradje je spisano v okolju Java, sicer pa programiramo s programskim jezikom na osnovi C/C++ [1].

Samo razvojno okolje je preprosto, kot je vidno na Sliki 2.5. Arduino ima za osnovo dve glavni metodi:

- Funkcija `Setup()` se izvede le prvič, ko se program zažene
- Funkcija `Loop()` se izvaja v zanki nepretrgoma



Slika 2.5: Razvojno okolje Arduino IDE



## Poglavje 3

# Razvoj strojne opreme

V tem poglavju je bolj podrobno opisan postopek povezave mikrokrmilnika Arduino in ELM 327 preko povezave Bluetooth z modulom HC-05, vezava LCD zaslona, SD bralnika in modula za realno uro.

### 3.1 Bluetooth modul HC-05

ELM 327 ima že vgrajen oddajnik Bluetooth signala, mikrokrmilnik Arduino pa še ne vsebuje niti oddajnika niti sprejemnika Bluetooth signala. Zato je treba dodati modul, ki omogoča povezavo Bluetooth. Odločili smo se za modul HC-05 (Slika 3.1), ki za komunikacijo uporablja serijsko UART komunikacijo, zato mikrokrmilnik Arduino z njim lahko komunicira. Modul smo na mikrokrmilnik povezali, kot piše v Tabeli 3.1.



Slika 3.1: BlueTooth modul HC-05

Tabela 3.1: Vezava BlueTooth modula

Modul	Arduino	Angleški opis	Slovenski opis	Posebnosti
GND	GND	Ground	zemlja	
VCC	5V	Power	napajanje	
RXD	TXD	Transmitting	oddajanje	
				Potrebna je vezava na
TXD	RXD	Receiving	sprejemanje	DC-DC pretvornik, ki zmanjša napetost iz 5V na 3.3V.

Pred začetkom je treba modul HC-05 najprej konfigurirati. Konfiguriranje poteka preko AT ukazov, ki mu jih posredujemo preko serijske UART povezave. AT ukazi so kratka serija znakov, ki imajo poseben pomen za napravo, ki jo nastavljamo [3]. Ime AT prihaja iz besede ATtention. Naprava loči dva različna načina delovanja:

1. Podatkovni način, v katerem naprava sprejema in oddaja podatke.

2. Ukazni (AT) način, v katerem naprava sprejema ukaze in oddaja odgovore.

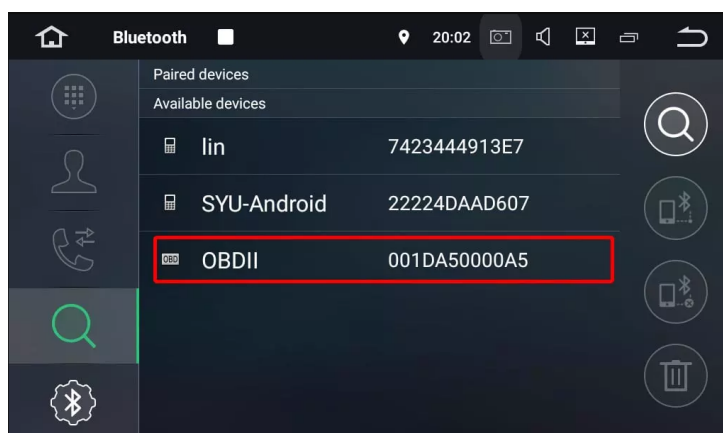
Modul je treba najprej spraviti v način AT. To storimo tako, da pritisnemo tipko na modulu, preden ga priklopimo. Ko je modul v načinu AT, začne LED lučka na njem utripati na vsaki 2 sekundi.

Zaporedje korakov, s katerimi smo konfigurirali modul HC-05:

1. AT; če nam modul na ta ukaz odgovori z OK, potem modul deluje.
2. AT+ORGL; ponastavi modul na tovarniške nastavitve.
3. AT+ROLE=1; s tem ukazom postavimo modul v funkcijo gospodarja.
4. AT+CMODE=0; modul se bo sedaj povezal s katero koli BlueTooth povezavo.
5. AT+BIND=MAC naslov vmesnika ELM327 konektorja; s tem ukazom povežemo z vmesnikom ELM327.

MAC naslov konektorja ELM 327 najlažje pridobimo tako, da konektor vklopimo v avto, na mobilnem telefonu pa vklopimo BlueTooth. Mobilni telefon prikaže vse naprave, ki oddajajo signal in njihov MAC naslov, kot na Sliki 3.2.

BlueTooth modul sprejema MAC naslove v obliki : XXXX:XX:XXXXXX. V zgornjem primeru je MAC naslov enak 00:1D:A5:00:00:A5, tako da bi morali za zadnji ukaz vtipkati AT+BIND=001D:A5:0000A5.



Slika 3.2: Iskanje MAC naslova vmesnika ELM 327

Po zaporedju zgornjih korakov je modul pripravljen za uporabo. Takoj, ko dobi napajanje, začne iskati BlueTooth signal, ki ga oddaja naprava s točno določenim MAC naslovom, in ko ga najde, se poveže z napravo.

## 3.2 LCD zaslon

Za izpis podatkov se uporablja standarden ekran za mikrokrmilnik Arduino velikosti 16\*2 znaka, na Sliki 3.4. LCD ima fizično nastavitve kontrasta slike in modro osvetlitev. LCD ekran se na mikrokrmilnik poveže z 10 žicami, vendar je na tem LCD zaslonu že integrirano vezje, ki s standardom I<sup>2</sup>C poenostavi nadzor nad zaslonom, tako da potrebujemo za povezavo samo 4 žice. LCD zaslon smo povezali, kot piše v Tabeli 3.2.

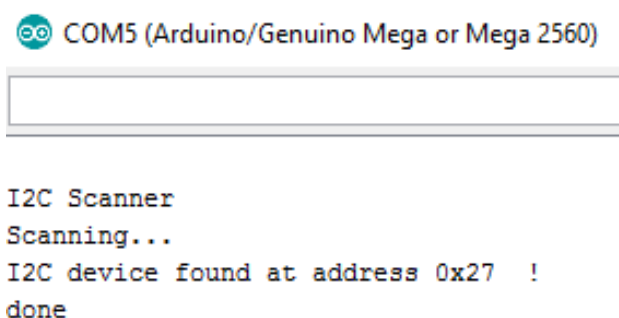
Tabela 3.2: Vezava LCD zaslona

Modul	Arduino	Angleški opis	Slovenski opis
GND	GND	Ground	zemlja
VCC	5V	Power	napajanje
SDA	Analog 1	Data	podatki
SCL	Analog 2	Clock Data	ura

## Protokol I<sup>2</sup>C

I<sup>2</sup>C je serijsko računalniško vodilo, ki ga je razvil Philips Semiconductor in je najpogostejša oblika komunikacije med integriranimi vezji. Vodilo je sestavljeno iz 2 žic; SDA je podatkovna žica, ki skrbi za prenos ukazov in odgovorov, SCL pa urin signal. Obe žici sta dvosmerni. Vsaka naprava, priključena na vodilo, mora imeti svoj enoličen naslov. Naslov je lahko vnaprej določen ali pa se spreminja. Komunikacijo vodi gospodar, ki sužnjem pošilja ukaze in podatke. Suženj lahko odgovori le takrat, ko mu to gospodar dovoli [13].

Da bi napravo lahko priključili na mikrokrmilnik preko standarda I<sup>2</sup>C, je treba najprej poznati njen enoličen naslov. Ker naslova ni bilo nikjer zapišanega, smo uporabili Arduino program [6], ki zazna vse I<sup>2</sup>C naprave, priključene na mikrokrmilnik ter izpiše njihove naslove Slika 3.3. V našem primeru je I<sup>2</sup>C naslov naprave 0x27.



```
COM5 (Arduino/Genuino Mega or Mega 2560)

I2C Scanner
Scanning...
I2C device found at address 0x27 !
done
```

Slika 3.3: Izhod programa I2C Scanner



Slika 3.4: LCD zaslon velikosti 16\*2

### 3.3 SD bralnik

V diplomski nalogi je predvidena uporaba SD kartice za shranjevanje parametrov. Mikrokrmilnik Arduino ne vsebuje bralnika SD kartic, zato je bilo treba uporabiti dodatni modul (Slika 3.5). Modul smo na mikrokrmilnik povezali, kot piše v Tabeli 3.3.



Slika 3.5: SD modul

Tabela 3.3: Vezava SD bralnika

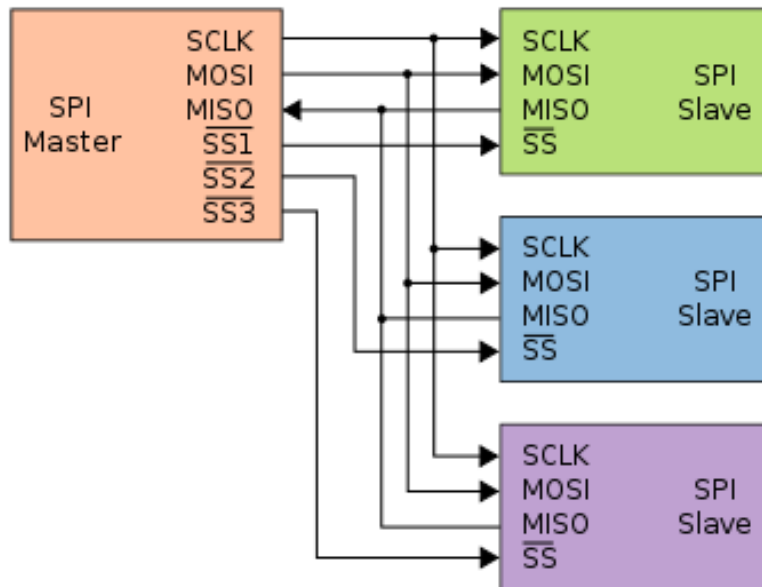
Modul	Arduino	Angleški opis	Slovenski opis
GND	GND	Ground	zemlja
VCC	5V	Power	napajanje
SS	9	Slave select	izbira sužnja
MOSI	10	SPI Data	podatki
MISO	11	SPI Data	podatki
SCK	12	SPI Clock	ura

### Protokol SPI

SD bralnik in mikrokrmilnik Arduino za komunikacijo uporabljata protokol SPI, ki je standard za sinhrono serijsko podatkovno povezavo elektronskih naprav in ki za komunikacijo uporablja konfiguracijo gospodar/suženj, vidno na Sliki 3.6. V našem primeru je mikrokrmilnik gospodar, SD modul pa suženj. Protokol omogoča tudi hkratno priključitev več sužnjev na enega gospodarja, pri čemer s signalom Slave Select (SS) določi, s katerim sužnjem komunicira [14].

Signali pri protokolu SPI:

- MISO (Master Input, Slave Output) - gospodar posluša, suženj sporoča;
- MOSI (Master Output, Slave Input) - gospodar sporoča, suženj posluša;
- SCK (Serial Clock) - ura, ki jo daje gospodar;
- SS (Slave Select) - s tem signalom gospodar določi, s katero napravo komunicira.



Slika 3.6: Povezava gospodar/suženj

### 3.4 Realna ura

V diplomski nalogi je uporabljeno shranjevanje parametrov na SD kartico v tekstovno datoteko. Zaradi kasnejšega pregledovanja in vizualizacije parametrov se je pojavila potreba po zapisu točnega časa, kdaj je bil parameter zajet. Mikrokrmilnik Arduino ne hrani točne ure zunanje sveta, zato je treba uporabiti modul za realno uro. Izbrali smo modul RTC DS1302 3.7, ki vsebuje baterijo CR2032 3V. Modul za realno uro je uporabljen za poimenovanje .txt datotek, v katere se shranjujejo parametri, in za kasnejše določanje točne ure branja posameznega parametra. Modul se je na mikrokrmilnik Arduino vezal, kot piše v Tabeli 3.4.



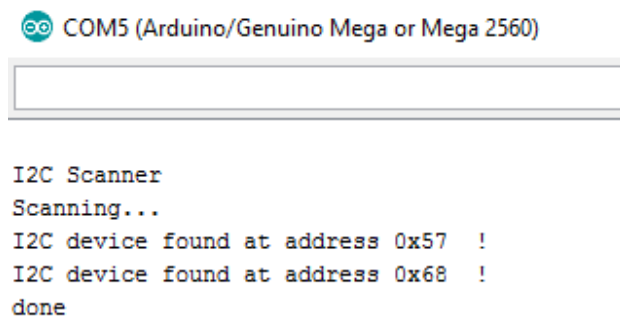
Tabela 3.4: Vezava modula za realno uro

Modul	Arduino	Angleški opis	Slovenski opis
GND	GND	Ground	zemlja
VCC	5V	Power	napajanje
SDA	Analog 1	Data	podatki
SCL	Analog 2	Clock Data	ura



Slika 3.7: Modul za hranjenje realne ure sistema

Modul za realno uro komunicira z mikrokrmilnikom preko protokola I<sup>2</sup>C ravno tako kot LCD zaslon, vendar to ne predstavlja ovire, saj je lahko na vodilo priključenih več naprav, če imajo vse različen naslov. Naslov naprave smo ravno tako pridobili z Arduino programom I2C Scanner, vidno na Sliki 3.8. V tem primeru vsebuje naprava poleg modula za realno uro še senzor temperature, ki ima ločen I<sup>2</sup>C naslov. S poizkušanjem smo ugotovili, da je pravi naslov realne ure 0x68.

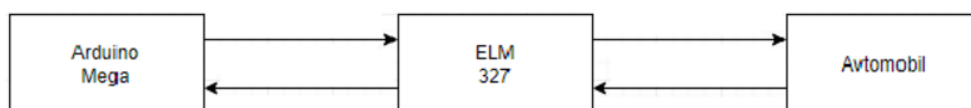


Slika 3.8: Izhod programa I2C Scanner

## Poglavje 4

# Razvoj programske opreme

V tem poglavju je predstavljeno programiranje v Arduino IDE, posamezni odseki bolj pomembne kode, predstavljen je format ukazov in odgovorov posameznih uporabljenih parametrov. Na Sliki 4.1 vidimo, kako poteka komunikacija med mikrokrmilnikom Arduino in motornim računalnikom v avtomobilu. Vmesnik ELM 327 deluje kot posrednik ukazov mikrokrmilnika motornemu računalniku ter nato posrednik odgovorov. Komunikacija med mikrokrmilnikom in vmesnikom ELM327 poteka preko signala BlueTooth.



Slika 4.1: Komunikacija med Arduino Mega in avtomobilom

### 4.1 Konfiguriranje vmesnika ELM 327

Mikrokrmilnik Arduino takoj po priklopu na napajanje začne iskati BlueTooth povezavo z našim vmesnikom ELM 327, ki ga prepozna po vnaprej določenem MAC naslovu. Ko se Arduino in vmesnik povežeta, lahko preko serijske povezave takoj posredujemo ukaze vmesniku ELM 327, ki jih posreduje naprej motornemu računalniku. Da smo se pravilno povezali, lahko

hitro ugotovimo, če dobimo na ukaz AT odgovor OK.

Preden lahko dostopamo do podatkov preko PID, je treba vmesnik ELM327 konfigurirati. To storimo z naslednjo kombinacijo ukazov:

1. ATZ - resetira vmesnik
2. ATSPX - izbere protokol
  - 0 - avtomatska zaznava protokala
  - 1 - SAE J1850 PWM (41,6 kbaud)
  - 2 - SAE J1850 VPW (10,4 kbaud)
  - 3 - ISO 9141-2 (5 baud init, 10,4 kbaud)
  - 4 - ISO 14230-4 KWP (5 baud init, 10,4 kbaud)
  - 5 - ISO 14230-4 KWP (fast init, 10,4 kbaud)
  - 6 - ISO 15765-4 CAN (11 bit ID, 500 kbaud)
  - 7 - ISO 15765-4 CAN (29 bit ID, 500 kbaud)
  - 8 - ISO 15765-4 CAN (11 bit ID, 250 kbaud)
  - 9 - ISO 15765-4 CAN (29 bit ID, 250 kbaud)

ELM 327 je po prejemu teh ukazov pripravljen na delovanje. Vmesnik sedaj po vpisu PID številke parametra vrne podatek, ki nas zanima.

## Prikaz razpoložljivih parametrov

Avtomobili se med seboj razlikujejo po številu in vrsti vgrajenih senzorjev in tako ni nujno, da lahko pri vseh beremo vse podatke, ki nam jih omogoča OBD2. Omejitve pri branju določenih parametrov ima lahko tudi uporaba poceni vmesnika ELM 327, ki ni sposoben brati toliko podatkov. Če napišemo ukaz za pridobivanje napačnega parametra ali pa tega parametra ne moremo brati, nam vmesnik javi ali NO DATA ali pa dobimo odgovor, ki se začne z bajtom 7F.

V izogib branju napačnih podatkov lahko s preprostim ukazom ugotovimo, katere parametre sploh lahko beremo. Za vsako področje podatkov je treba poslati svoj ukaz in ga dekodirati. Vsi ukazi so enake oblike, in sicer najprej številka področja v šestnajstiškem sestavu, nato pa dva znaka 0. Protokol OBD2 deli podatke na naslednja področja:

- 0100 - trenutni podatki,
- 0200 - prikaz shranjenih podatkov,
- 0300 - prikaz shranjenih diagnostičnih napak,
- 0400 - izbris shranjenih podatkov in napak,
- 0500 - testiranje in prikaz rezultatov ter spremljanje kisika,
- 0600 - testiranje in prikaz rezultatov ter ostalih komponent,
- 0700 - prikaz shranjenih diagnostičnih napak med zadnjo vožnjo,
- 0800 - nadzor delovanja sistemov v avtomobilu,
- 0900 - prikaz splošnih podatkov v avtomobilu,
- 0A00 - prikaz pobrisanih napak.

Vsi ukazi, ki jih uporabljamo pri nalogi, so iz področja 0100 torej trenutni podatki.

### **Primer dekodiranja razpoložljivih parametrov**

Vsi odgovori so šestnajstiški, zato jih je treba najprej kodirati v dvojiški številski sestav, ki pa neposredno pove, ali je parameter podprt (1) ali ne (0). Primer dekodiranja za področje 1 je naveden v Tabeli 4.1.

- Ukaz: 0100
- Odgovor: CA178078

Tabela 4.1: Parametri v področju 1

Šestnajstiško	Dvojiško	Podpora	PID	Angleški opis
C	1	Da	0101	Monitor status
	1	Da	0102	Freeze DTC
	0	Ne	0103	Fuel system status
	0	Ne	0104	Calculated engine load
A	1	Da	0105	Engine coolant temperature
	0	Ne	0106	Short term fuel trim—Bank 1
	1	Da	0107	Long term fuel trim—Bank 1
	0	Ne	0108	Short term fuel trim—Bank 2
1	0	Ne	0109	Long term fuel trim—Bank 2
	0	Ne	010A	Fuel pressure
	0	Ne	010B	Intake manifold absolute pressure
	1	Da	010C	Engine RPM
7	0	Ne	010D	Vehicle speed
	1	Da	010E	Timing advance
	1	Da	010F	Intake air temperature
	1	Da	0110	MAF air flow rate
8	1	Da	0111	Throttle position
	0	Ne	0112	Commanded secondary air status
	0	Ne	0113	Oxygen sensors present
	0	Ne	0114	Oxygen Sensor 1
0	0	Ne	0115	Oxygen Sensor 2
	0	Ne	0116	Oxygen Sensor 3
	0	Ne	0117	Oxygen Sensor 4
	0	Ne	0118	Oxygen Sensor 5
7	0	Ne	0119	Oxygen Sensor 6
	1	Da	011A	Oxygen Sensor 7
	1	Da	011B	Oxygen Sensor 8
	1	Da	011C	OBD standards this vehicle conforms to
8	1	Da	011D	Oxygen sensors present
	0	Ne	011E	Auxiliary input status
	0	Ne	011F	Run time since engine start
	0	Ne	0120	PIDs supported [21 - 40]

Avtomobil, ki bi na ukaz 0100 vrnil tak odgovor, podpira naslednje parametre: 0101, 0102, 0105, 0107, 010C, 010E, 010F, 0110, 0111, 011A, 011B, 011C in 011D. Za nepodpiranje ostalih je lahko kriv avto, ki nima senzorjev ali pa ELM 327 ne podpira teh ukazov.

## 4.2 Seznam uporabljenih parametrov

V diplomski nalogi so uporabljeni parametri, ki so navedeni v Tabeli 4.2.

Tabela 4.2: Uporabljeni parametri

PID	Število vrnjenih bajtov	Opis	Formula za izračun	Enota
0104	1	Obremenjenost motorja	$\frac{100}{255} * A$	%
0105	1	Temperatura hladilne tekočine	$A - 40$	$^{\circ}C$
010C	2	Obrati motorja	$\frac{256*A+B}{4} * A$	rpm
010D	1	Hitrost vozila	$A$	$\frac{km}{h}$
010B	1	Tlak v sesalnem kolektorju	$A$	kPa
0110	2	Pretok zraka	$\frac{256*A+B}{100} * A$	$\frac{g}{s}$
0111	1	Položaj pedala za plin	$\frac{100}{255} * A$	%
0121	1	Prevoženi kilometri z MIL	$256 * A + B$	km

### 4.2.1 Primer izračuna obremenjenosti motorja

Ukaz za pridobitev podatka o obremenjenosti motorja je v avtomobilu podprt in vrne en bajt podatkov, ki jih je potrebno pretvoriti v desetiški številski sestav ter uporabiti pravilno enačbo iz Tabele 4.2.

- Ukaz : 0104
- Odgovor : 41 04 35

4	1	04	35
Pravilnost odgovora	OBD2 področje	Parameter	A

Odgovor nam pove, da je bil ukaz uspešen (4), da je to odgovor na ukaz s področja 1, navezuje pa se na parameter 04, ki je obremenjenost motorja. Ukaz nam vrne samo en bajt podatkov, in to je 35. Podatek je v šestnajstiškem številskem sestavu, zato ga je treba pred uporabo pretvoriti v desetiški številski sestav.

$$35_{(16)} = 53_{(10)}$$

Za pravi podatek o obremenjenosti motorja je treba upoštevati še enačbo

$$\frac{100}{255} * A = \frac{100}{255} * 53 = 20,78\% \quad (4.1)$$

Obremenjenost motorja je trenutno 20,78 %.

#### 4.2.2 Primer izračuna obratov motorja

Ukaz za pridobitev podatka o obratih motorja je v avtomobilu podprt in vrne dva bajta podatkov, ki jih je potrebno pretvoriti v desetiški številski sestav ter uporabiti pravilno enačbo iz Tabele 4.2.

- Ukaz : 010C
- Odgovor : 41 0C 14 25

4	1	0C	14	25
Uspešnost ukaza	OBD2 področje	Parameter	A	B

Odgovor nam pove, da je bil ukaz uspešen (4), da je to odgovor na ukaz s področja 1, navezuje pa se na parameter 0C, kar pomeni obrati motorja. Ukaz 010C nam vrne dva bajta podatkov, 14 in 25. Podatka sta v šestnajstiškem številskem sestavu, zato ju je treba najprej pretvoriti v desetiški številski sestav.

$$14_{(16)} = 20_{(10)}$$



$$25_{(16)} = 37_{(10)}$$

Obrati motorja se sedaj izračunajo po enačbi:

$$\frac{A * 256 + B}{4} = \frac{256 * 20 + 37}{4} = 1289,25 \frac{\text{obrat}}{\text{min}} \quad (4.2)$$

### 4.3 Izračun trenutne porabe

Izračun trenutne porabe temelji na predpostavkah o idealnem mešanju zraka in nafte ter gostoti nafte, zato izračun ni popolnoma zanesljiv. Celotna enačba za izračun porabe je povzeta po viru [7].

Za izračun trenutne porabe avtomobila potrebujemo podatke iz Tabele 4.3.

Tabela 4.3: Izračun trenutne porabe

Ime	Kratica	Formula	Enota
Mešanica zraka, potrebnega za dizelski motor	MRAD	14,5	
Gostota nafte	DFD	0,83	$\frac{kg}{l}$
Pretoka zraka	MAF	PID	$\frac{g}{s}$
Hitrost vozila	SP	PID	$\frac{km}{h}$
Masni pretok goriva	MFF	$\frac{MAF * 3600}{MRAD}$	$\frac{kg}{s}$
Volumenski pretok goriva	VFF	$MFF * DFD$	$\frac{l}{h}$
Trenutna poraba goriva	FC	$\frac{VFF * 100}{SP}$	$\frac{l}{100km}$

#### 4.3.1 Primer izračuna trenutne porabe

Za izračun potrebujemo dva parametra iz avtomobila, in sicer podatek o hitrosti in o pretoku zraka. Podatkov zaradi počasnih standardov pri nekaterih vozilih ne moremo pridobivati istočasno, zato je treba med pridobivanjem dveh podatkov počakati najmanj 500 ms.

Ukaz za pridobivanje hitrosti:

- Ukaz: 010D

- Odgovor: 41 0D 53

Odgovor je pravilen, kar vidimo po prvem znaku (4), odgovor je na ukaz o trenutni hitrosti avtomobila (10D), vrednost parametra pa je 53. Parameter pretvorimo v desetiški številski sestav in to je tudi že realna hitrost, saj je formula za izračun tega parametra enaka vrnjeni številki.

Izračunana hitrost je  $53 \frac{km}{h}$ .

Ukaz za pridobivanje podatkov o pretoku zraka:

- Ukaz: 0110
- Odgovor: 41 10 08 FE

Odgovor je pravilen, kar vidimo po prvem znaku (4), odgovor je na ukaz pretoku zraka (104), vrednost parametra pa je  $A = 8$ ,  $B = 254$ . Parameter pretvorimo v desetiški številski sestav in pravilen pretok zraka izračunamo po formuli:

$$\frac{256 * A + B}{100} = \frac{256 * 8 + 254}{100} = 23,02 \frac{kg}{s}. \quad (4.3)$$

Izračunani pretok zraka je  $23,02 \frac{kg}{s}$ .

Z izračunanima podatkom o trenutni hitrosti in pretoku zraka, ki smo ju pridobili prek vmesnika OBD2, ter z vnaprej določeno gostoto nafte in razmerjem mešanja goriva z zrakom lahko izračunamo trenutno porabo. Primer izračuna:

$$MRAD = 14,5$$

$$DFD = 0,83$$

$$MAF = 23,02 \frac{g}{s} = 0,02302 \frac{kg}{s}$$

$$MFF = \frac{MAF * 3600}{MRAD} = \frac{0,02302 * 3600}{14,5} = 5,7153 \frac{kg}{h}$$

$$VFF = MFF * DFD = 5,7153 * 0,83 = 4,7437 \frac{l}{h}$$

$$\text{Trenutna poraba goriva} = \frac{VFF * 100}{\text{Hitrost}} = \frac{4,7437 * 100}{53} = 8,95 \frac{l}{100km}$$

## 4.4 Pomembni deli kode

### 4.4.1 Nastavitev realne ure

Modulu za realno uro je treba najprej nastaviti začetno uro, datum in dan v tednu, šele nato je sposoben ohranjati uro. Začetna ura se nastavi s kodo na Sliki 4.2.

```
void setDS3231time(byte second, byte minute, byte hour,
    byte dayOfWeek, byte dayOfMonth, byte month, byte year)
{
    Wire.beginTransmission(DS3231_I2C_ADDRESS);
    Wire.write(0);
    Wire.write(decToBcd(second));
    Wire.write(decToBcd(minute));
    Wire.write(decToBcd(hour));
    Wire.write(decToBcd(dayOfWeek));
    Wire.write(decToBcd(dayOfMonth));
    Wire.write(decToBcd(month));
    Wire.write(decToBcd(year));
    Wire.endTransmission();
}
```

Slika 4.2: Nastavljanje ure

Uro nastavimo za klicanje funkcije setDS323time [10] (parameter 1, parameter 2, parameter 3, parameter 4, parameter 5, parameter 6, parameter 7), pri čemer so parametri:

1. sekunde,
2. minute,
3. ura,
4. dan v tednu,
5. dan,
6. mesec,

7. leto.

#### 4.4.2 Ukaz in odgovor ELM 327

Parametri se zaradi počasnosti protokolov OBD2, sploh pri starejših vozilih, lahko berejo vsakih 500 ms, vsa komunikacija pa poteka tako, da vtipkamo ukaz in mu dodamo znak za novo vrstico `\r`. Nato čakamo na odgovor in sestavljamo odgovor, dokler se ne pojavi znak `>`, ki pomeni, da se je odgovor končal. Posredovanje ukazov prikazuje Slika 4.3.

```
//FUNCTION FOR READING PARAMETERS
void read_parameter(String parameter,int write_out){
    command = parameter + "\r";
    inputString = "";
    Serial3.print(command);
    while(Serial3.available()){
        recvChar = (char)Serial3.read();
        if(recvChar != ' '){
            inputString += recvChar;
        }
        if (recvChar == '>') {
            s = inputString;
            break;
        }
    }
}
```

Slika 4.3: Zapis ukaza in branje odgovora

#### 4.4.3 Shranjevanje podatkov na SD kartico

Podatka o številu obratov na minuto in hitrosti se vsakih 5 sekund zapišeta na SD kartico, z namenom nadzorovanja voznika. Podatke shranjujemo na kartico v formatu CSV, kar nam olajša uvoz podatkov v Excel, kjer podatke lažje preučujemo. Podatke shranjujemo v datoteko, poimenovano po dnevu, v katerem smo vozili, poleg tega pa za vsak zapis hitrosti in obratov shranjujemo še točno uro zapisa (to nam omogoča modul za realno uro). Shranjevanje podatkov prikazuje Slika 4.4.

```
String times = displayTime();  
myFile = SD.open(year_name, FILE_WRITE);  
if (myFile) {  
    myFile.println(times+ "," + String(speed_test) + "," + String(revs));  
}  
else{  
    Serial.println("ERROR");  
}  
myFile.close();
```

Slika 4.4: Shranjevanje podatkov na SD kartico

#### 4.4.4 Seznam uporabljenih knjižnic

V projektu so bile z namenom hitrejšega dela uporabljene knjižnice z naslednjimi zaglavnimi (header) datotekami:

- Wire.h [20], za delo z modulom za realno uro.
- LCD.h [9], za delo z zaslonom.
- LiquidCrystal\_I2C.h [8], za delo s protokolom  $I^2C$ .
- SPI.h [18], za delo s protokolom SPI.
- SD.h [17], za shranjevanje podatkov na SD kartico.



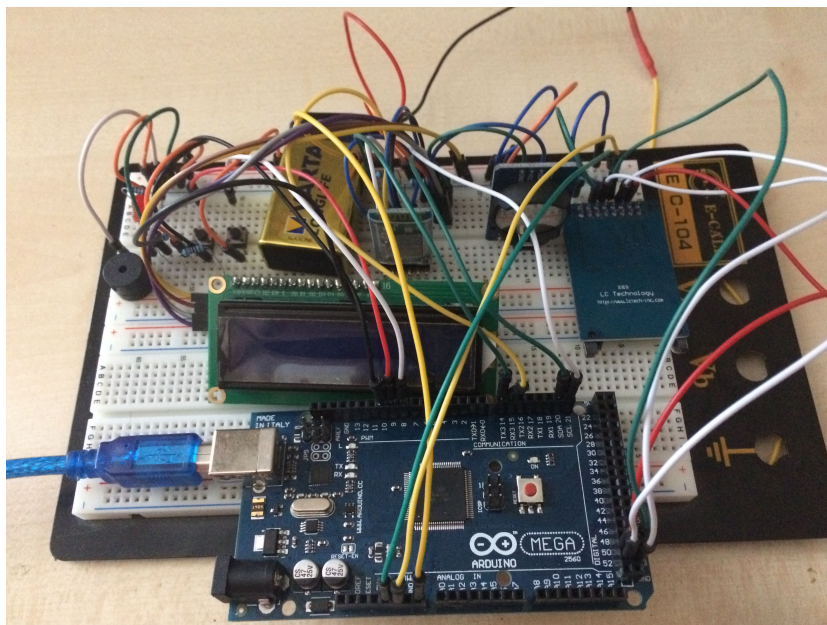
## Poglavje 5

# Testiranje

V tem poglavju je predstavljeno testiranje končnega izdelka na več avtomobilih različnih znamk in modelov. Predstavljeni so rezultati podpiranja parametrov različnih avtomobilov ter videz končnega izdelka.

### 5.1 Končni izdelek

Končni izdelek podpira branje parametrov ter prikaz na zaslonu LCD. Podatki se osvežujejo na 500 ms, ob tem pa se podatki o hitrosti, obratih in trenutni uri zapisujejo na kartico SD za kasnejši pregled na računalniku. S pritiskom na gumb se sprehajamo po prikazu različnih parametrov. Podprti so tisti parametri, ki jih je možno prebrati na vseh testiranih avtomobilih. Uporabljeni parametri so navedeni v Tabeli 4.1.



Slika 5.1: Izdelek med testiranjem

Cena končnega produkta:

- Arduino Uno 25€[5]
- Arduino Mega 45€[4]
- ELM 327 15€
- LCD 15€
- Bluetooth modul 15€
- SD kartica in SD modul 10€

Skupaj: 125€

Skupna cena je, v primerjavi z zastonsko aplikacijo za mobilni telefon, visoka, vendar v primerjavi z namenskimi orodji za diagnosticiranje avtomobilov zelo nizka.



### 5.1.1 Seznam uporabljenih parametrov

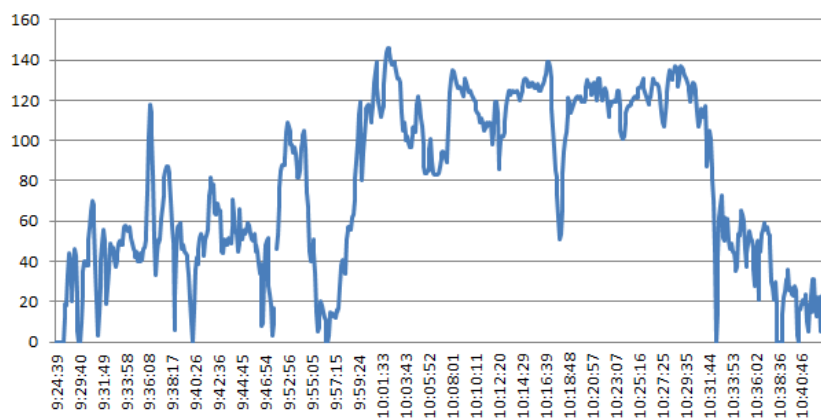
1. Trenutna poraba
2. Povprečna poraba
3. Obrati na minuto
4. Trenutna hitrost
5. Obremenjenost motorja
6. Temperatura hladilne tekočine
7. Tlak v sesalnem kolektorju
8. Pretok zraka
9. Položaj pedala za plin
10. Število prevoženih kilometrov z lučko MIL

## 5.2 Testiranje na avtomobilih

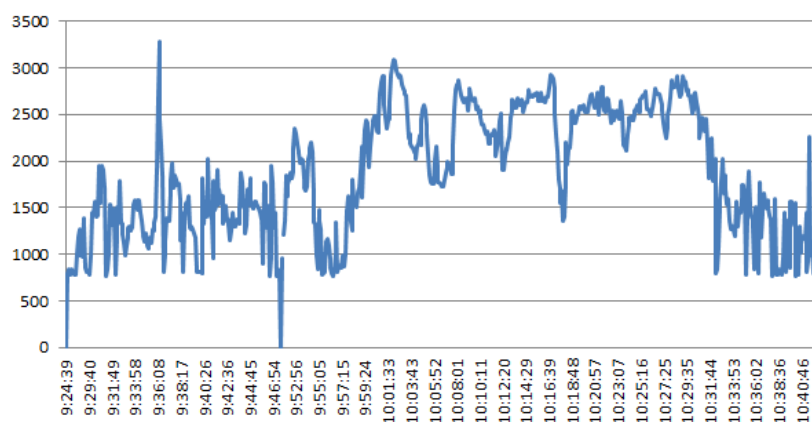
Razvoj je potekal na vozilu Volkswagen Golf, letnik 2005. V fazi testiranja nas je zanimalo, če bi izdelek deloval enako tudi na vozilih drugih znamk, ter, ali podpira več parametrov. Podatka o hitrosti in obratih sta se vsakih 5 sekund zapisala na SD kartico v datoteko s končnico CSV. Za vsak dan posebej se ustvari nova datoteka, v katero pišemo z imenom trenutnega datuma. Datoteko lahko odpremo s programom Excel, kot na Sliki 5.2. Med testiranjem je bila z avtomobilom Volkswagen Golf opravljena pot iz Škofje Loke do Novega mesta, iz nastalih podatkov pa sta nastala graf hitrosti na Sliki 5.3 in graf obratov motorja na Sliki 5.4.

9:24:39	0	0
9:24:43	0	799
9:24:49	0	832
9:27:49	0	799
9:27:55	0	783
9:28:01	0	832
9:28:07	0	815
9:28:14	0	815
9:28:20	13	783
9:28:26	19	783
9:28:32	18	1024
9:28:38	33	1199
9:28:44	44	1263
9:28:50	39	991
9:28:57	26	1216
9:29:03	20	975
9:29:09	38	1391
9:29:15	46	1295
9:29:21	43	847
9:29:27	23	815
9:29:34	5	815
9:29:40	0	799
9:29:46	0	783
9:29:52	9	991
9:29:58	24	1439

Slika 5.2: Shranjevanje podatkov na SD kartico



Slika 5.3: Graf hitrosti



Slika 5.4: Graf obratov motorja

### 5.2.1 Volkswagen Golf, 1,9 TDI, letnik 2005

V avtomobilu Volkswagen Golf dobimo pri ukazu, ki nam pove, katere parametre podpira OBD2 protokol, odgovor: 41 00 98 3B 80 11 v šestnajstiškem številskem sestavu. Odgovor nam pove, da je bil ukaz uspešen (4), da je to odgovor na ukaz s področja 1, navezuje pa se na parameter 00, kar pomeni razpoložljivost parametrov. Ko ostale bajte (98 3B 80 11) pretvorimo v dvojiški številski sestav dobimo 1001 1000 0011 1011 1000 0000 0001 0001. Odgovor nam tako pove, da Golf podpira parametre 1, 4, 5, 11, 12, 13, 15, 16, 17, 28, 32. Skupno število podprtih parametrov je 11.

### 5.2.2 Audi A4, 1,9 TDI, letnik 2003

Avtomobil Audi A4 nam, kljub temu da je 2 leti starejši, vrne na ukaz o podprtih parametrih 41 00 98 3B 80 11, kar pomeni, da podpira iste parametre kot Golf.

### 5.2.3 Volkswagen Polo, 1,2 TSI, letnik 2011

Pri avtomobilu Volkswagen Polo smo že vnaprej predvidevali, da bo število podprtih parametrov večje kot pri starejših dveh avtomobilih. Na ukaz o

podprtih parametroh smo dobili številko 41 00 BE 3E A8 13. Odgovor nam pove, da je bil ukaz uspešen (4), da je to odgovor na ukaz s področja 1, navezuje pa se na parameter 00, kar pomeni razpoložljivost parametrov. Ko ostale bajte (BE 3E A8 13) pretvorimo v dvojiški številski sestav dobimo 1011 1110 0011 1110 1010 1000 0001 0011. Avtomobil torej podpira parametre: 1, 3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 17, 19, 21, 28, 31, 32. Skupno število podprtih parametrov je 17.

## Poglavje 6

### Sklepne ugotovitve

Namen diplomske naloge je bil raziskovanje protokola OBD2 ter razvoj izdelka na osnovi mikrokrmilnika Arduino in vmesnika ELM 327, ki bi komuniciral z računalnikom v avtomobilu, kar nam je tudi uspelo. Hoteli smo podpreti čim več parametrov, saj OBD2 podpira veliko različnih. V fazi testiranja smo ugotovili, da so parametri odvisni od avtomobila in števila vgrajenih senzorjev v njem, tako da smo podprli tiste, ki delujejo v vseh testiranih avtomobilih. Med razvijanjem programske opreme je bilo treba zaradi predolge programske kode zamenjati mikrokrmilnik Arduino Uno za Arduino Mega, ki ima več pomnilniškega prostora.

Izdelek je v fazi delovanja in deluje solidno. Težave so z BlueTooth modulom, ki se občasno noče povezati, z resetiranjem modula pa se težava odpravi. Če bi se diplomske naloge lotil še enkrat, bi izbral kakovostnejši modul. Priказani parametri, ki se dajo preveriti, se ne razlikujejo veliko.

Možne izboljšave in dopolnitve so shranjevanje vseh parametrov, podpora prikazovanja porabe tudi za avtomobile z bencinskim motorjem, na podporo parametrov pa, žal, ne moremo vplivati.

Tema diplomske naloge je zanimiva in aktualna, saj gre avtomobilska prihodnost v smer, ki vsebuje vedno več elektronskih komponent in računalniških sistemov, kar počasi vodi k avtonomni vožnji. Avtomobil bo zbiral in obdeloval vse več podatkov o vožnji, ki jih bodo lahko brali le proizvajalci in

avtomobilski servis, ki bo imel originalno napravo, ki je narejena specifično za znamko avtomobila. S tem pa pride na plano tudi sama varnost avtomobila in potnikov v njem, saj je lažje nadzorovati in vdreti v programsko opremo kot pa fizično ukrasti avto. Napadi na programsko opremo v avtomobilu so se že zgodili preko brezžičnih povezav; raziskovalcem je že leta 2010 uspelo onеспособiti zavore in ugasniti motor [19]. Znana je ranljivost avtomobilov znamke BMW, ko tat s fizično silo najprej vstopi v avtomobil nato pa prek protokola OBD2 ukrade podatke o ključu, jih naloži na svoj ključ ter odpelje avtomobil [16].







# Literatura

- [1] Arduino programski jezik. Dosegljivo: <https://www.arduino.cc/en/Reference/HomePage>. [Dostopano: 24. 5. 2017].
- [2] Arduino: shranjevanje programa in spremenljivk. Dosegljivo: <https://www.arduino.cc/en/Tutorial/Memory>. [Dostopano: 13. 7. 2017].
- [3] AT ukazi. Dosegljivo: <https://www.techopedia.com/definition/575/at-command-set>. [Dostopano: 25. 8. 2017].
- [4] Cena Arduino Mega. Dosegljivo: <https://www.sparkfun.com/products/11061>. [Dostopano: 30. 7. 2017].
- [5] Cena Arduino Uno. Dosegljivo: <https://www.sparkfun.com/products/11021>. [Dostopano: 30. 7. 2017].
- [6] I2C Scanner. Dosegljivo: <https://playground.arduino.cc/Main/I2cScanner>. [Dostopano: 13. 7. 2017].
- [7] Izračun porabe. Dosegljivo: <https://wiki.52north.org/Projects/DieselConsumptionCalculation>. [Dostopano: 13. 7. 2017].
- [8] LCD I2C knjižnica. Dosegljivo: <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>. [Dostopano: 26. 7. 2017].
- [9] LCD knjižnica. Dosegljivo: <https://github.com/nherment/Arduino-Library/blob/master/LCD/LCD.h>. [Dostopano: 26. 7. 2017].

- 
- [10] Nastavljanje ure. Dosegljivo: <http://tronixstuff.com/2014/12/01/tutorial-using-ds1307-and-ds3231-real-time-clock-modules-with-arduino/>. [Dostopano: 26. 7. 2017].
- [11] OBD2. Dosegljivo: [https://en.wikipedia.org/wiki/On-board\\_diagnostics](https://en.wikipedia.org/wiki/On-board_diagnostics). [Dostopano: 24. 5. 2017].
- [12] Primerjanje mikrokontrolerov Arduino. Dosegljivo: <https://www.arduino.cc/en/Products/Compare>. [Dostopano: 13. 7. 2017].
- [13] Protokol I2C. Dosegljivo: <https://www.svet-el.si/o-reviji/samogradnje/835-i2c-monitor>. [Dostopano: 13. 7. 2017].
- [14] Protokol SPI. Dosegljivo: <https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee3081/datasheets/S12SPIV3.pdf>. [Dostopano: 13. 7. 2017].
- [15] Protokoli OBD2. Dosegljivo: [http://www.obdtester.com/obd2\\_protocols](http://www.obdtester.com/obd2_protocols). [Dostopano: 13. 7. 2017].
- [16] Ranljivost v BMW. Dosegljivo: <http://www.extremetech.com/extreme/132526-hack-the-diagnostics-connector-steal-yourself-a-bmw-in-3-minutes>. [Dostopano: 26. 8. 2017].
- [17] SD knjižnica. Dosegljivo: <https://github.com/adafruit/SD/blob/master/SD.h>. [Dostopano: 26. 7. 2017].
- [18] SPI knjižnica. Dosegljivo: <https://github.com/PaulStoffregen/SPI>. [Dostopano: 26. 7. 2017].
- [19] Varnost v avtomobilih. Dosegljivo: <https://www.eset.com/si/podjetje/novice/article/7-stvari-ki-jih-morate-vedeti-o-hekanju-avtomobilov/>. [Dostopano: 27. 7. 2017].
- [20] WIRE knjižnica. Dosegljivo: <https://github.com/esp8266/Arduino/tree/master/libraries/Wire>. [Dostopano: 26. 7. 2017].

- 
- [21] Matic Končan. Aplikacija za optimizacijo sistema varčne vožnje vozil na motorni pogon. Diplomski nalogi, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013.
- [22] OBD2 projekt. Dosegljivo: <http://www.eecs.ucf.edu/seniordesign/sp2011su2011/g09/Documents/finalpaper.pdf>. [Dostopano: 3.5. 2017].
- [23] Aplikacija Torque. Dosegljivo: <https://play.google.com/store/apps/details?id=org.prowl.torque&hl=sl>. [Dostopano: 24.5. 2017].